Very good job! Passed.

732A38 - Computational statistics - Lab 4

Group 6: Araya Eamrurksiri & Oscar Pettersson

29 february 2016

Assignment 1: Computations with Metropolis-Hastings

Consider the following probability density function:

$$f(x) \propto x^5 e^{-x}, x > 0$$

You can see that the distribution is known up to some constant of proportionality.

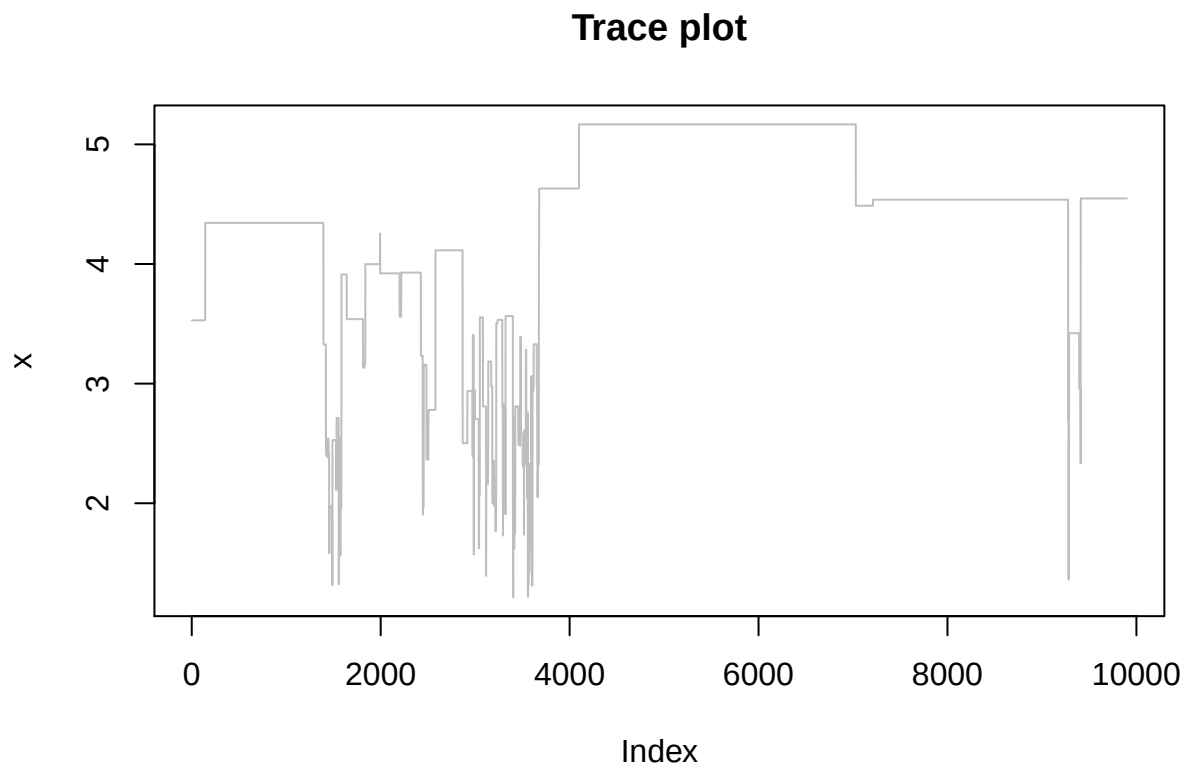
1.1 Use Metropolis-Hastings algorithm to generate samples from this distribution by using proposal distribution as log-normal $LN(X_t, 1)$, take some starting point. Plot the chain you obtained as a time series plot. What can you guess about the convergence of the chain? If there is a burn-in period, what can be the size of this period?

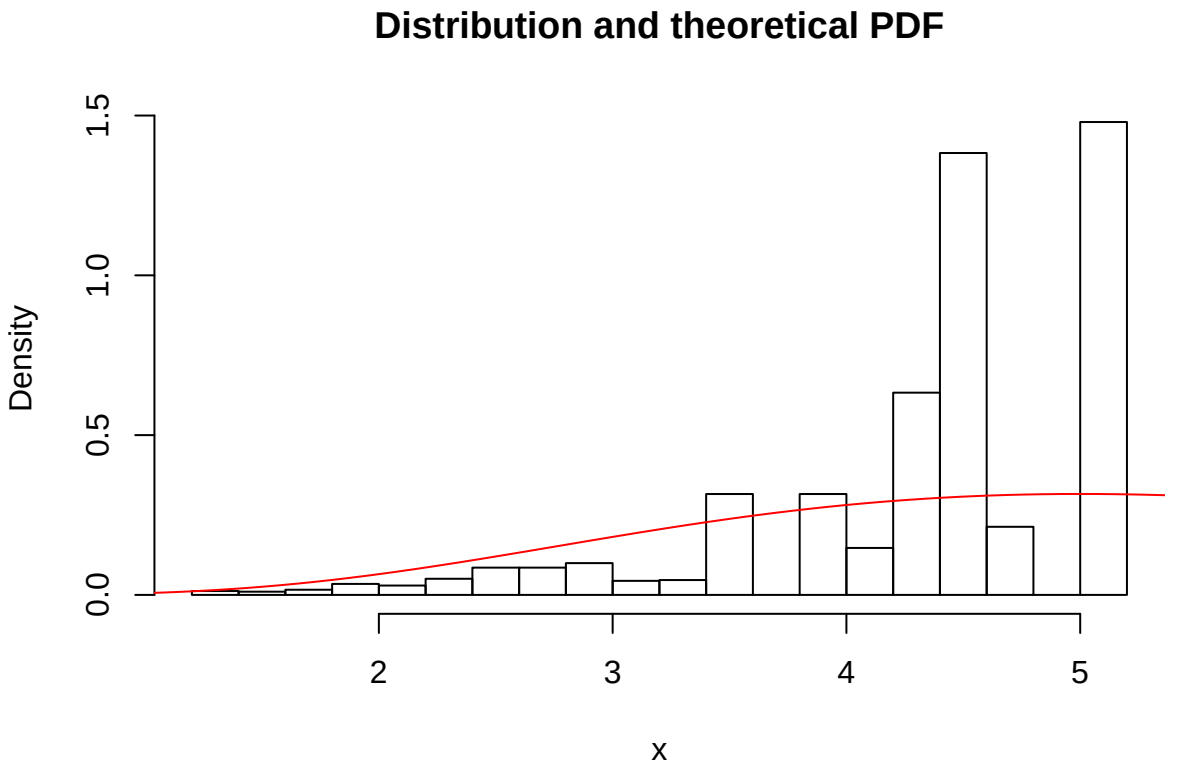
We want to sample data from an unknown distribution by using Markov Chain Monte Carlo methods. We know that the PDF $f(x)$ is proportional to $x^5 e^{-x}$, for $x > 0$. We use the Metropolis-Hastings algorithm and 10,000 iterations, with the log-normal$(X_t, 1)$ distribution as proposal distribution and a random logN(0, 1) sample as a starting point.

```
MCMC <- function(n, pi., q., gen_q, X0 = runif(1), ...) {
  x <- rep(NA_real_, n)
  x[1] <- X0 # 1. Starting point X_0
  for(t in 2:n) {
    Y <- gen_q(n = 1, x[t - 1], ...) # 2. Candidate point Y
    u <- runif(1) # 3. Random u from U(0, 1)
    frac1 <- pi.(Y) * q.(x = x[t - 1], Y, ...)
    frac2 <- pi.(x[t - 1]) * q.(x = Y, x[t - 1], ...)
    alpha <- min(1, frac1 / frac2)

    if(u <= alpha) { # 4. Comparison of u and alpha(X_t, Y)
      x[t] <- Y
    } else {
      x[t] <- x[t - 1]
    }
  } # 5. Next loop
  return(x)
}
```

```
f <- function(x) exp(-x) * x ^ 5
set.seed(12345)
sigma <- 1
res_1 <- MCMC(n = 1E4, pi. = f, q. = dlnorm, gen_q = rlnorm, X0 = rlnorm(1), sdlog = sigma)
MCMC_plot(result = res_1[-c(1:100)], pi. = f, c = 0.015, xmin = 0, xmax = 6)
```


Trace plot
x
2
3
4
5
0
2000
4000
6000
8000
10000
Index

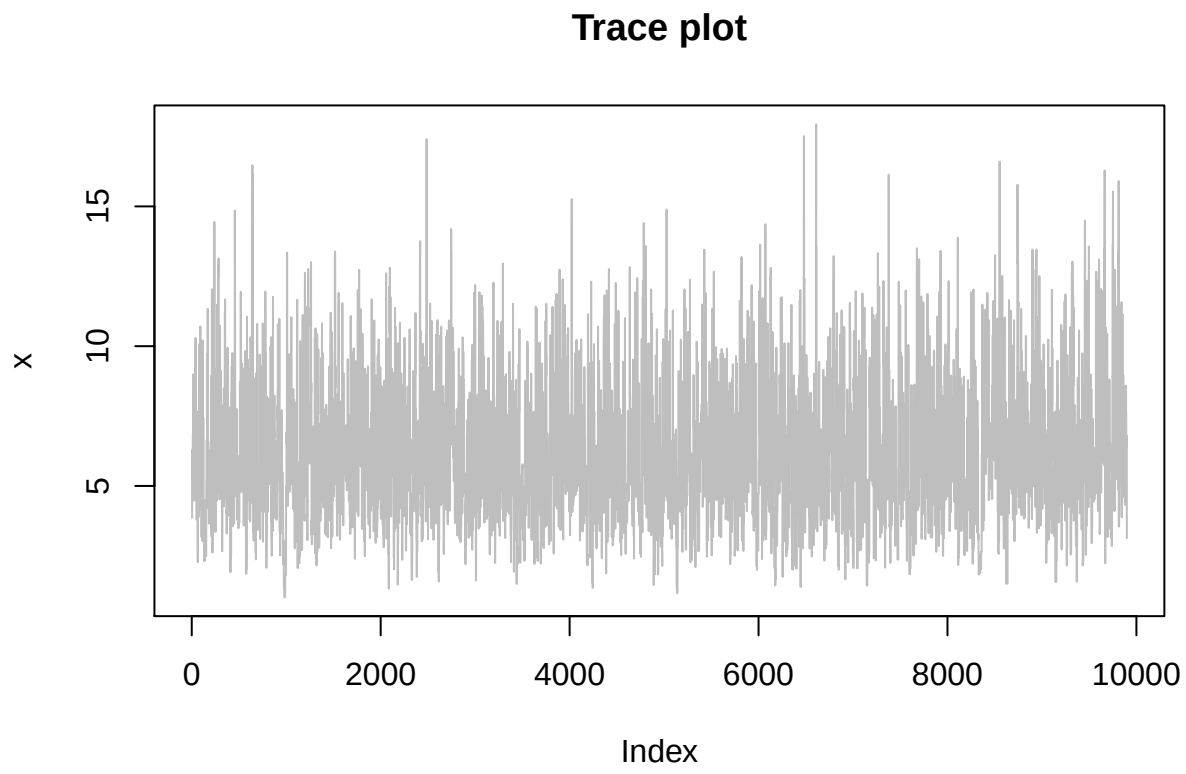

Distribution and theoretical PDF
Density
0.0
0.5
1.0
1.5
2
3
4
5
x

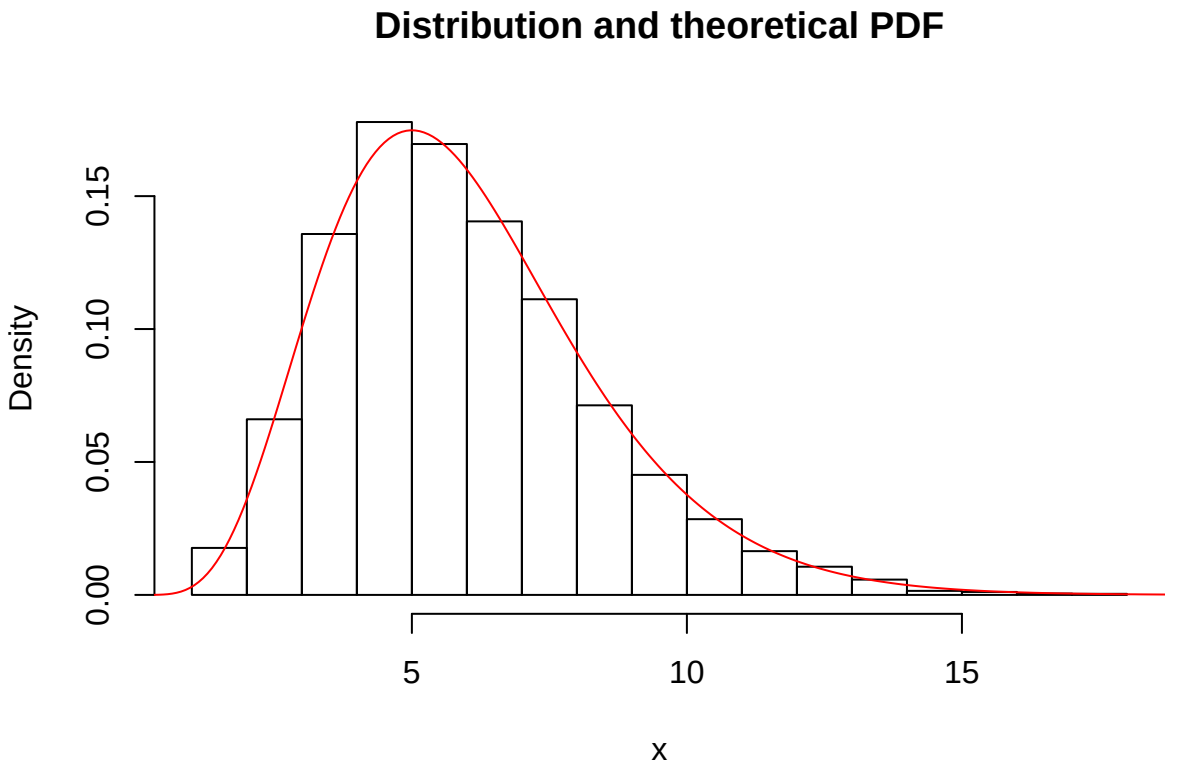
We can see in the trace plot above that there are many places where the chain does not move and it doesn't look like this chain will converge for a larger number of iterations. Thus, there is no burn-in period. The distribution is not very good either since it does not resemble the theoretical PDF very well.

1.2 Perform step 1 by using chi-square distribution $\chi^2\left(floor\left(X_t+1\right)\right)$ as proposal distribution where $floor(x)$ means integer part of x.

Now, a χ^2-distribution where the degrees of freedom are decided by $floor(X_t+1)$ is used as a proposal distribution instead. The initial point is a random $\chi^2(2)$ sample.

```
dchisq_floor <- function(x, df) dchisq(x = x, df = floor(df))
rchisq_floor <- function(n, df) rchisq(n = n, df = floor(df))
set.seed(12345)
res_2 <- MCMC(n = 1E4, pi. = f, q. = dchisq_floor, gen_q = rchisq_floor, X0 = rchisq_floor(1, 2))
MCMC_plot(result = res_2[-c(1:100)], pi. = f, c = 0.0083, xmin = 0, xmax = 20)
```


Trace plot
x
5
10
15
0
2000
4000
6000
8000
10000
Index


Distribution and theoretical PDF
Density
0.00
0.05
0.10
0.15
5
10
15
x

This chain looks much more like white noise around a constant mean and the burn-in period is maybe only a few iterations. And the distribution of the observations (excluding the first 100) looks very much more like the theoretical one.

1.3 Compare the results of steps 1 and 2 and make conclusions.

The figure of the chain in step 1.1 shows that it does not converge to the target distribution. The reason for that might be because of poorly mixing. On the other hand, the chain in step 1.2 seems stationary as there is no obvious trend or change in spread. This chain has a good mixing and a very good convergence.

Therefore, using chi-square as proposal distribution seems to be a better choice to obtain samples from the given distribution rather than using log-normal distribution. This is because the χ^2-distribution is a special case of the Gamma distribution so the second proposal distribution is more close to the target ditto than is the log-normal.

1.4 Generate 10 MCMC sequences using the generator from the step 2 and with starting points 1,2,.., or 10. Use Gelman-Rubin method to analyze convergence of these sequences.

Now, ten MCMC-sequences are generated with the χ^2 distribution as proportional distribution, as before. The starting points X_0 are 1, 2, ..., 10. Then, the Gelman-Rubin method is used, via the coda package, to investigate whether they converged or not.

```
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## [1,]          1          1
```

Since the upper bound of the Gelman-Rubin factor $\sqrt{\hat{R}}$ is about 1, we don't believe that the estimated variance of e.g. the mean of the sequences will change if we add more iterations and we conclude that the chains have converged.

1.5 Estimate $\int_0^\infty x \cdot f(x)dx$ using the samples from steps 1 and 2.

In order to estimate the integral $\int_0^\infty x \cdot f(x)dx$ for steps 1 and 2, we realise that this is the expected value of X, since $x > 0$. So, we estimate this by the sample mean. For the second chain, we first remove the burn-in period, though.

```
print(mean(res_1))
```

```
## [1] 4.379952
```

```
print(mean(res_2[-c(1:100)]))
```

```
## [1] 5.935091
```

So, the means are not so close to each other for the two chains.

1.6 The distribution generated is in fact a gamma distribution. Look in the literature and define the actual value of the integral. Compare it with the one you obtained.

We can see that $f(x)$ is a special case of a Gamma distribution. A Gamma distribution has $f(x) = \frac{\lambda e^{-\lambda x}(\lambda x)^{t-1}}{\Gamma(t)}$ [Martínez, Martínez] and here, $\lambda = 1$ and $t = 6$. The expected value (the value of the integral) is $\frac{t}{\lambda}$ which is 6 here and about the estimate for when the χ^2-distribution was used as a proposal distribution but not so close to the first run. This confirms the previous results.

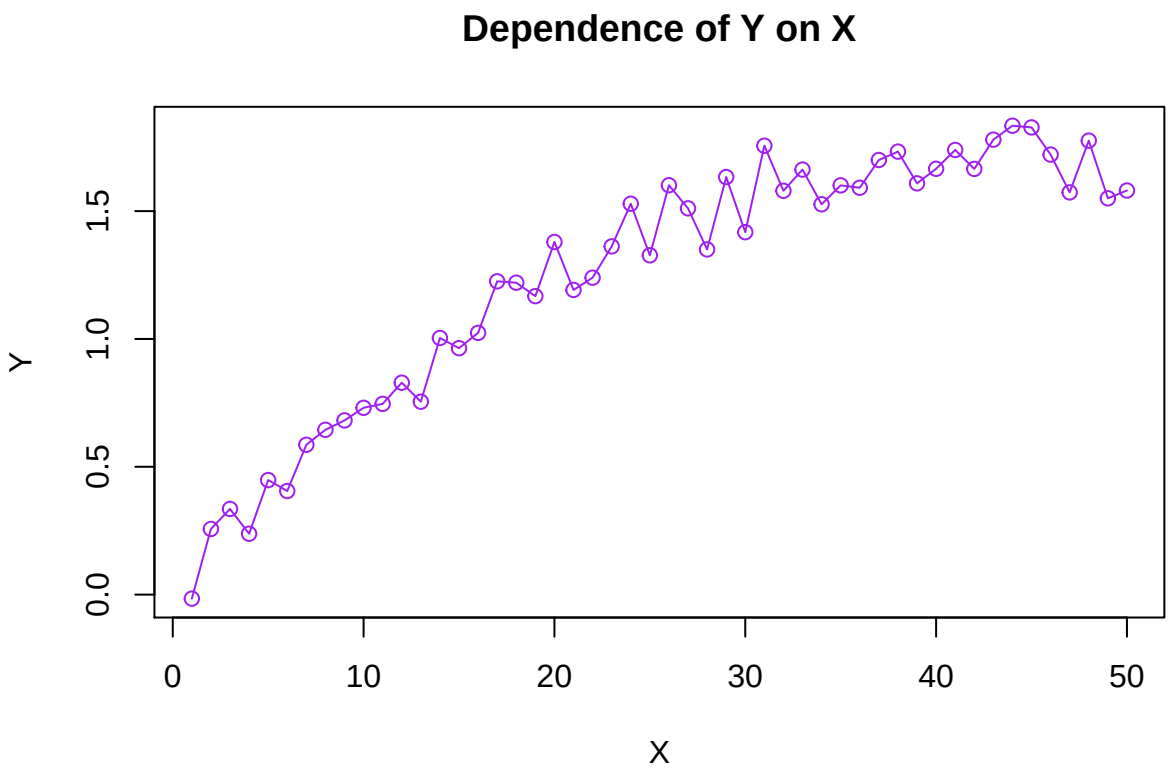
Assignment 2: Gibbs sampling

A concentration of a certain chemical was measured in a water sample, and the result was stored in the data chemical.RData having the following variables:

- X: day of the measurement
- Y: measured concentration of the chemical.

The instrument used to measure the concentration had certain accuracy; this is why the measurements can be treated as noisy. Your purpose is to restore the expected concentration values.

2.1 Import the data to R and plot the dependence of Y on X. What kind of model is reasonable to use here?



An analysis of the graph gives that the value moderately increase over time with some fluctuation. It can be seen that linear model probably will result in bad fits. A model that can capture a nonlinear pattern is therefore thought to be needed in this case.

2.2 A researcher has decided to use the following (random walk) Bayesian model (n=number of observations, $\mu = (\mu_1, ..., \mu_n)$ are unknown parameters):

$$Y_i \sim N(\mu_i, var = 0.2), i = 1, ..., n$$

where the prior is $p(\mu_1) = 1$ and $\mu_{i+1} \sim N(\mu_i, 0.2), i = 1, ..., n-1\$$

Present the formulas showing the likelihood $p(Y|\mu)$ and the prior $p(\mu)$ (hint: a chain rule can be used here $p(\mu) = p(\mu_1)p(\mu_2|\mu_1)p(\mu_3|\mu_2...p(\mu_n|\mu_{n-1})))$

- Likelihood $p(Y|\mu)$:

$$p(Y|\mu) = (0.4\pi)^{-\frac{n}{2}} exp(-\frac{1}{0.4}\sum_{i=1}^{n}(Y_i - \mu_i)^2)$$

- Prior $p(\mu)$:

$p(\mu_2|\mu_1) = \frac{1}{\sqrt{0.4\pi}} exp(-\frac{1}{0.4}(\mu_2 - \mu_1)^2)$

...

$p(\mu_n|\mu_{n-1}) = \frac{1}{\sqrt{0.4\pi}} exp(-\frac{1}{0.4}(\mu_n - \mu_{n-1})^2)$

From the chain rule, we get

$$p(\mu) = \frac{1}{(0.4\pi)^{\frac{n-1}{2}}} exp(-\frac{1}{0.4}\sum_{i=2}^{n}(\mu_i - \mu_{i-1})^2)$$

2.3 Use the Bayes theorem to get the posterior up to a constant of proportionality, and then find out the distributions for $\mu_i|\mu_{-i}, Y$ where μ_{-i} is a vector containing all μ values except of μ_i

- Posterior $p(\mu|Y)$:

$= p(Y|\mu) \cdot p(\mu)$

$= \frac{1}{(0.4\pi)^{n-\frac{1}{2}}} exp[-\frac{1}{0.4}(\sum_{i=2}^{n}(\mu_i - \mu_{i-1})^2 + \sum_{i=1}^{n}(Y_i - \mu_i)^2)]$

- Distribution for $\mu_i|\mu_{-i}, Y$:

$p(\mu_1|\mu_{-1}, Y) \propto exp[-\frac{(\mu_1 - \frac{\mu_2+Y_1}{2})^2}{2(0.1)}]$

$\Rightarrow N(\frac{\mu_2+Y_1}{2}, 0.1)$

$p(\mu_i|\mu_{-i}, Y) \propto exp[-\frac{(\mu_i - \frac{\mu_{i-1}+\mu_{i+1}+Y_i}{3})^2}{\frac{2(0.2)}{3}}]; i = 2, 3, ..., n-1$

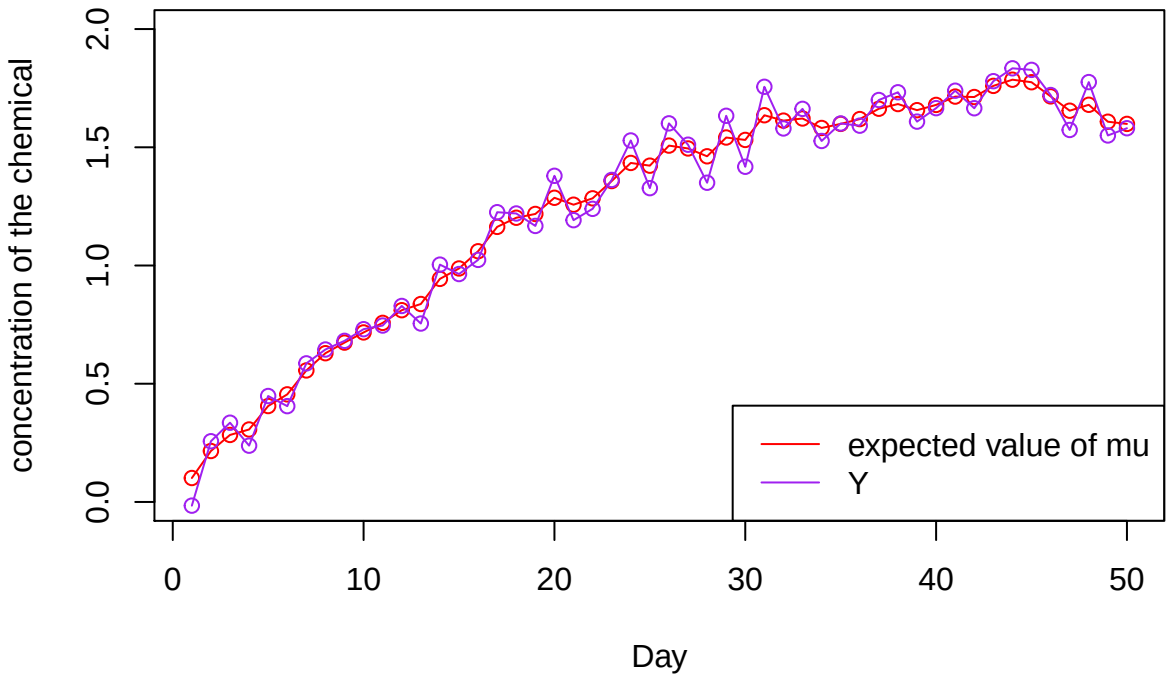
$\Rightarrow N(\frac{\mu_{i-1}+\mu_{i+1}+Y_i}{3}, \frac{0.2}{3})$

$p(\mu_n|\mu_{-n}, Y) \propto exp[-\frac{(\mu_n - \frac{\mu_{n-1}+Y_n}{2})^2}{2(0.1)}]$

$\Rightarrow N(\frac{\mu_{n-1}+Y_n}{2}, 0.1)$

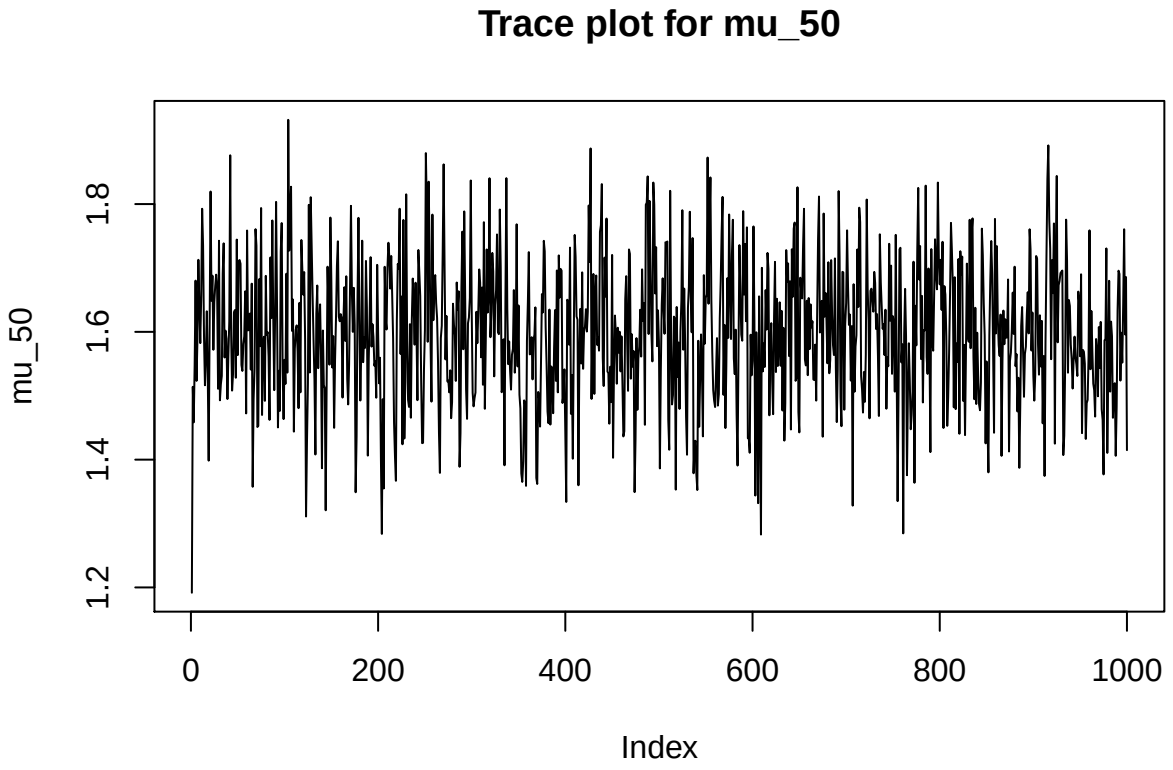
2.4 Use the distributions derived in step 3 to implement a Gibbs sampler that uses $\mu^0 = (0, ..., 0)$ as a starting point. Run the Gibbs sampler to obtain 1000 values of μ and then compute the expected value of μ by using Monte Carlo approach. Plot the expected value of μ versus X and Y versus X in the same graph. Does it seem that you have managed to remove the noise? Does it seem that the expected value of μ can catch the true underlying dependence between Y and X?

The following is a plot of the expected value of μ versus X and also Y versus X:



The red line indicates the expected value of μ while the purple line represents the value of Y in the dataset. The expected value of μ is computed by discarding the first 20 values as it is the size of burn-in period (obtained after some visual inspection). From looking at this plot, it can be seen that the noise have been removed from the data as the line looks smoother. Moreover, the expected value of μ seems to be able to catch the true underlying dependence between Y and X.

2.5 Make a trace plot for μ_{50} and comment on the burn-in period and convergence.



The figure above shows the trace plot for μ_{50} which also represents a good mixing. The chain for μ_{50} is converged and looking at the plot, we can see that the burn-in period is very short.

Contribution

For the first assignment, the code and most of the text are from Oscar's report and for the second assignment, the code and the text are from Araya's report. Other than that, only small changes and mutually written parts have been made. There were no big differences between the reports.

R-code

```
MCMC <- function(n, pi., q., gen_q, X0 = runif(1), ...) {
  x <- rep(NA_real_, n)
  x[1] <- X0 # 1. Starting point X_0
  for(t in 2:n) {
    Y <- gen_q(n = 1, x[t - 1], ...) # 2. Candidate point Y
    u <- runif(1) # 3. Random u from U(0, 1)
    frac1 <- pi.(Y) * q.(x = x[t - 1], Y, ...)
    frac2 <- pi.(x[t - 1]) * q.(x = Y, x[t - 1], ...)
    alpha <- min(1, frac1 / frac2)

    if(u <= alpha) { # 4. Comparison of u and alpha(X_t, Y)
      x[t] <- Y
    } else {
      x[t] <- x[t - 1]
    }
  } # 5. Next loop
  return(x)
}
MCMC_plot <- function(result, pi., c, xmin, xmax, trace_plot = TRUE, histogram = TRUE) {
  if(trace_plot) plot(result, type = "l", col = "gray", main = "Trace plot", ylab = "x")
  cat("\n")
  if(histogram) {
    hist(result, probability = TRUE, main = "Distribution and theoretical PDF", xlab = "x")
    x_theo <- seq(from = xmin, to = xmax, length.out = 250)
    y_theo <- c * pi.(x_theo)
    points(y_theo ~ x_theo, type = "l", col ="red")
  }
}
f <- function(x) exp(-x) * x ^ 5
set.seed(12345)
sigma <- 1
res_1 <- MCMC(n = 1E4, pi. = f, q. = dlnorm, gen_q = rlnorm, X0 = rlnorm(1), sdlog = sigma)
MCMC_plot(result = res_1[-c(1:100)], pi. = f, c = 0.015, xmin = 0, xmax = 6)
dchisq_floor <- function(x, df) dchisq(x = x, df = floor(df))
rchisq_floor <- function(n, df) rchisq(n = n, df = floor(df))
set.seed(12345)
res_2 <- MCMC(n = 1E4, pi. = f, q. = dchisq_floor, gen_q = rchisq_floor, X0 = rchisq_floor(1, 2))
MCMC_plot(result = res_2[-c(1:100)], pi. = f, c = 0.0083, xmin = 0, xmax = 20)
library(coda)
res_4 <- matrix(data = NA_real_, nrow = 1E4, ncol = 10)
for(X0 in 1:10) {
  set.seed(12345)
  res_4[, X0] <- MCMC(n = 1E4, pi. = f, q. = dchisq_floor, gen_q = rchisq_floor, X0 = X0)
  # MCMC_plot(result = res_4[, X0], pi. = f, c = 0.008, xmin = 0, xmax = 20)
}
MCMC_list <- mcmc.list()
for (i in 1:10) MCMC_list[[i]] <- as.mcmc(res_4[,i])
gelman.diag(MCMC_list)
print(mean(res_1))
print(mean(res_2[-c(1:100)]))
load("chemical.RData")
```

```r
plot(X,Y,type="o",col="purple", main="Dependence of Y on X")
gibbs <- function(Y,sample){
    mu.list <- matrix(ncol=50, nrow=sample)
    mu <- rep(0, 50)
    for(j in 1:sample){
        for(i in 1:50){
            if(i == 1){
                mu[i] <- rnorm(1, mean=( (mu[i+1]+Y[i])/2 ), sd=0.1)
            }else if(i == 50){
                mu[i] <- rnorm(1, mean=( (mu[i-1]+Y[i])/2 ), sd=0.1)
            }else{
                mu[i] <- rnorm(1, mean=( (mu[i-1]+mu[i+1]+Y[i])/3 ), sd=0.2/3)
            }
        }
        mu.list[j,] <- mu
    }
    return(mu.list)
}

ans <- gibbs(Y,1000)

#expected value
e.mu <- apply(ans[21:1000,],2,mean)
plot(X, e.mu, type='o', col='red', ylim=c(0,2), xlab="Day", ylab="concentration of the chemical")
points(X,Y, type='o', col='purple')
legend("bottomright", c("expected value of mu","Y"), col=c("red","purple"), lwd=1)
plot(ans[,50], type='l', ylab="mu_50", main="Trace plot for mu_50")
## NA
```